

# FanGrower: A simple bi-resolution triangle-mesh

Ramswaroop Somani and Jarek Rossignac  
College of Computing and GVU Center  
Georgia Institute of Technology  
{somani,jarek}@cc.gatech.edu}

## Abstract

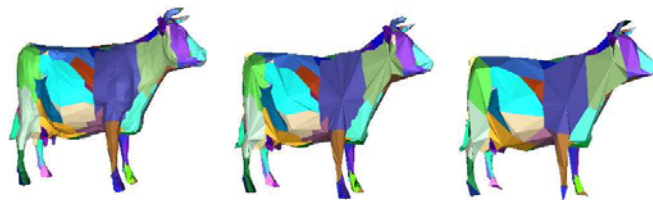
The FanGrower algorithm proposed here segments a manifold triangle mesh into regions (called caps), which may each be closely approximated by a triangle-fan. Once the caps are formed, their rims, which form the inter-cap boundaries, are simplified, replacing each fan by its frame—a fan with the same apex but fewer triangles. The resulting collection of frames is an approximation of the original mesh with a guaranteed maximum error bound. As such, it may be viewed as a powerful extension of Kalvin and Taylor’s super-faces, which were restricted to nearly planar configurations and approximated by nearly planar fans. In contrast, our caps simplify to frames that need not be planar, but may contain convex or concave corners or saddle points. We propose a new and efficient solution for evaluating a tight bound on the deviation between a cap and its approximating fan and frame. We also introduce a new solution for computing the location of the apex of a fan as the point minimizing Garland and Heckbert’s quadric error for a set of planes defined by the vertices of the cap and their normals. We discuss several cap-growing approaches. Finally, we propose a compact representation of a triangle mesh from which one can easily extract the frames and execute selective refinements needed to reconstruct the original caps in portions of the mesh that are closer to the viewer, to a silhouette, or in an area of interest. Some frames are automatically upgraded to partly simplified fans to ensure a water-tight transition between frames and application-selected caps.

{**Keywords:** Triangle-meshes, Simplification, Error-estimation, Multi-Resolution Modeling}

## Introduction

Many applications in Computer Graphics manipulate complex 3D models. The models are often described by an approximating triangle mesh represented by the coordinates of its vertices and by triplets of indices defining triangle/vertex incidence. In spite of considerable progress in 3D compression and graphic acceleration, the number of triangles required to produce good approximations of complex shapes limits the transmission and rendering speed. Numerous approaches have been proposed to simplify triangle meshes. They reduce the number of vertices in the model by either grouping triangles into a relatively small number of nearly flat regions of which the interior vertices may be omitted or by collapsing groups of adjacent or neighboring vertices, so as to reduce their number without perturbing too much the shape of the 3D model. We describe here a new simplification approach, called **FanGrower**, which groups triangles into simply connected regions, called **caps**. The caps need not be flat and may take

the form of a more general shape, as long they may be closely approximated by a triangle fan whose boundary, called **rim**, coincides with the boundary of the cap. Given the rim of a cap, its **fan** is completely defined by its apex, a point called the **tip**: the fan is the set of triangles that join the tip and the rim. Note that the cap and its fan have the same boundary. By simplifying the rims, we reduce the number of triangles in a fan. We ensure that such a simplification preserves the connectivity graph between caps. We use the term **frame**, to denote a simplified fan. The caps of a cow model, with their fans and frames are shown in Fig. 1.



**Fig 1:** The 5804 triangle mesh of the cow model was divided by into 168 color-coded caps (left). The fan mesh (center) contains 2274 triangles. The frames (right) contain a total of 914 triangles. The caps, fans, and frames were generated by FanGrower in 27 seconds.

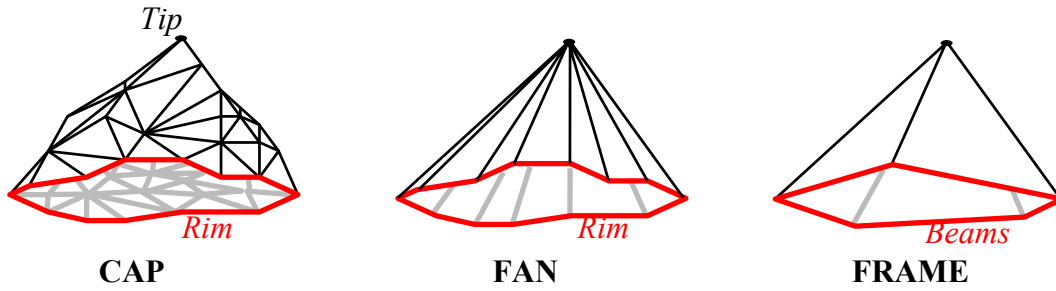
One may use the triangle mesh formed by the triangles of all the frames as a single resolution simplification of the original mesh. However, when more resolution is desired in some parts of the model, an automatically selected subset of the frames may be replaced by the original caps. To prevent cracks and T-junctions, a frame cannot be adjacent to a cap. We use partly simplified fans to ensure a water-tight transition between caps and frames.

The three stages in the evolution of a cap and our terminology are illustrated symbolically in Fig. 2. The details of the actual caps, fans, and frames produced by FanGrower on a typical 3D model are shown in Fig. 3.

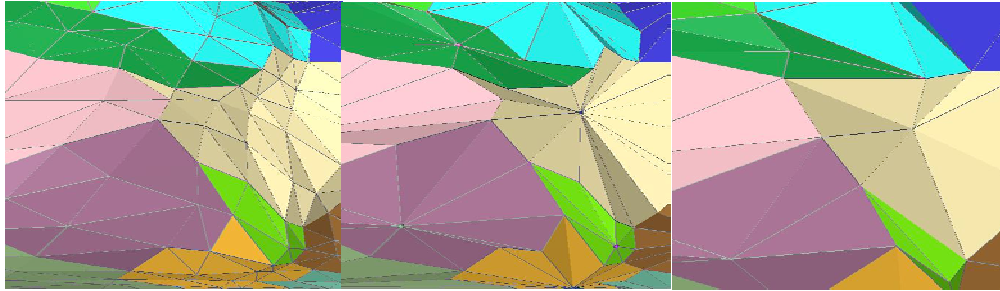
FanGrower grows one cap at a time. The triangles that have not yet been absorbed in a cap are called the **virgin** triangles. The cap-growing process involves the following steps:

**Identify seed-tip:** The isolation measure associates with each triangle the average graph-distance to other virgin triangles. We select the most isolated triangle as the initial cap and its barycenter as the initial tip.

**Grow caps:** Virgin triangles that share an edge with the current cap are tested and possibly included in the cap. The test ensures that the discrepancy (Hausdorff distance) between a cap and its fan does not exceed a given discrepancy threshold and that the rim of the cap satisfies some shape constraints upon which we



**Fig 2:** A cluster of triangles (left) and its fan (center) joining the boundary (rim) of the cluster to its tip (apex of the fan). A simplified fan (right) was obtained by simplifying the boundary of the cluster.



**Fig 3:** The caps (left), their fans (center), and their frames (right).

rely when computing the bound on the discrepancy. The inclusion of a new triangle in the cap may occasionally require moving the tip to a new location, attempting to minimize the discrepancy.

**Fit beams:** Each connected chain of edges that all bound the same pair of caps are replaced by a straight line segments between its end-vertices. Then the line segment is refined adaptively by recursively splitting it at the vertex where the error between the original chain and its simplification is the largest. The splitting process stops when the largest error falls below the desired threshold. We use the term **beam** to denote each one of the edges of this simplified representation. With each beam, we associate a representation of its portion of the initial chain.

**Build a fan-mesh:** A simple triangle-mesh data structure is used to represent the mesh formed by all of the triangles joining the tip of each cap with its beams. These triangles form the frame of the cap. The tips and beams are each associated with a one-bit refinement mask and with geometry and connectivity upgrades, from which partial fans and caps may be restored.

For simplicity, our discussion and implementation is restricted to manifold (and water-tight) triangle meshes or to pseudo-manifold representations that may have been algorithmically derived from triangulations of the boundaries of non-manifold solids [RoCa99].

The paper is organized as follows. We first review prior art in triangle-mesh simplification. Then, we explain various implementation details: a technique for limiting the undulations of the rim, an approach for computing a tight bound on the maximum deviation between a cap and its fan and frame, and a fast solution for computing a good location of the tip. Then, we discuss clustering approaches and

report experimental results. Finally, we discuss a compact representation of triangle meshes that factors out their frames from the details, which are represented as upgrades to frames and of their rim-edges.

## Prior art in triangle-mesh simplification

Simplification techniques have been surveyed in [Blak87, HeGa94, HeGa97, PuSc97, Ross95, Ross96, Ross00] and more recently in [Lu&02]. Most approaches segment the vertices or triangles of a triangle mesh into features and simplify each feature independently without exceeding the allowed error bound or in an order that attempts to minimize some error estimate. They differ in the size or nature of the features they can handle, in the techniques used for identifying the features, in the techniques used to simplify the individual features, and in the technique for estimating the resulting error.

Turk treated an entire surface as a single feature, uniformly distributing new vertices on the surface and removing old ones while preserving the original topology [Turk92]. Other researchers chose to compute a tolerance zone around the original surface and fit a simple mesh inside that zone [Vars94, MiSu95, An&96, Co&96]. He et al. [He&96] rasterize the original surface into a volume, perform a low-pass 3D filter transformation to eliminate details, and then compute a new triangle mesh as a level set of the filtered volume.

Rossignac and Borrel's **vertex clustering** approach [RoBo93] perform a crude vertex quantization (which uses an axis aligned grid and declares that all the vertices that fall in the same grid cell form a cluster), select a representative vertex for each cluster, and eliminate degenerate triangles that have more than one vertex in the same cluster or replace them by a dangling edge or vertex, where appropriate. Low and Tan [LoTa97] have used

floating cells to make vertex clustering less susceptible to produce artifacts resulting from a particular alignment of the cell boundaries.

Although vertex clustering simplification provides a guaranteed upper bound on the Hausdorff error (i.e., the length of the diagonal of a cell), it rarely offers the most accurate simplification for a given triangle-count reduction, because the size of the cluster is fixed and cannot expand to span arbitrarily large and nearly flat regions. To enable such an expansion, Hoppe [Ho&93,Hopp96,Hopp98] and Ronfard and Rossignac [RoRo96] have independently devised progressive simplification techniques that collapse edges one-by-one in the order that minimizes the resulting error. (Both solutions use a priority queue to maintain a list of potential edge-collapses sorted by increasing error estimate.) Each edge-collapse operation eliminates one edge, one vertex and two triangles, and modifies neighboring triangles. In fact, the edge-collapse is equivalent to the preferred case [Schr97] of the vertex decimation approach proposed by Shroeder, Zarge, and Lorensen [Sc&92], who identify nearly flat triangle fans, remove their tip, and re-triangulate the resulting hole.

Note that each edge collapse merges two vertex clusters, but that vertex clusters formed by edge-collapses are restricted to be edge-connected, while those formed by Rossignac and Borrel's vertex clustering approach need not be connected, thus making it possible to simplify the topology of a shape, merging components or closing through-holes. Popovic and Hoppe [PoHo97] and Garland and Heckbert [GaHe97] have proposed to combine the edge-collapses [RoRo96,Hopp96] and vertex clustering [RoBo93] approaches into a single process.

Although the FanGrower simplification proposed here indirectly identifies edge-connected clusters of vertices, it does not do so through a series of edge-collapses. Instead, it follows the cluster growing strategy of Kalvin and Taylor [KaTa96], who compute "superfaces", which are features made by nearly coplanar clusters of triangles, and replace each feature by a star-shape triangulation with a single vertex at the center of the feature. Furthermore, Kalvin and Taylor simplify the boundaries between features by merging nearly collinear edges. The main difference between their approach and FanGrower lies in the restrictions imposed on the vertex clusters and in the techniques for estimating the error between the original and the simplified model.

## Prior art on error estimation

Most popular simplification techniques perform a sequence of edge-collapse operations, striving to select, at each stage, the edge whose collapse will have the smallest impact on the total error between the resulting simplified mesh and the original surface. Thus, they associate with each potential edge-collapse an error estimate and maintain a priority queue which allows to efficiently identify the best edge-collapse at each stage. Deciding how to estimate the error is delicate. The cost of computing it must be controlled, because the errors associated with the potential collapse of a given edge are impacted by the prior-collapses of neighboring edges, and thus may need to be reevaluated multiple times.

Some attempts at estimating the error that results from using approximations to nominal shapes for 3D graphics were based on objective or subjective measures focused on preserving image fidelity [Lu&02, Lind00], including the location of silhouettes or highlights [Hopp97, LuEr97]. Others were focused on measuring or estimating view-independent geometric deviations [RoBo93, KaTa96, Co&96, GaHe97]. For example, Cohen et al. [Co&96] have used Varshney's tolerance envelopes [Vars94], while Xia and Varshney [XiVa96] have used the sum of the length of collapsed edges as an error measure for an edge-collapse simplification process.

The maximum deviation,  $E(A,B)$ , between a shape  $A$  and a shape  $B$  may be formulated as the maximum of the distance,  $d(p,S)$ , from all points  $p$  on  $A$  or  $B$ , to  $S$ , which denotes the other shape, respectively  $B$  or  $A$ . This formulation is equivalent to the Hausdorff distance  $H(A,B)$ , which may also be formulated as the smallest radius  $r$ , such that  $A \subset B \hat{+} r$  and  $B \subset A \hat{+} r$ , where  $S \hat{+} r$  denotes the expanded set  $S$  obtained by replacing each point  $q$  of  $S$  by a ball of center  $q$  and radius  $r$ ; or equivalently by adding to  $S$  all points that are within distance  $r$  from it. The distance  $d(p,S)$  may be computed as the minimum of the distances between  $p$  and the following entities: (1) the vertices of  $S$ , (2) the normal projections of  $p$  onto the edges of  $S$ , and (3) the normal projections of  $p$  onto the interiors of the triangles of  $S$ . The difficulty in computing  $H(A,B)$  lies primarily in the fact that it is not sufficient to test  $d(p,S)$  for all vertices  $p$  of  $A$  and  $B$ , because the maximum discrepancy may occur inside a face. Consequently, the exact Hausdorff measure is often approximated by super-sampling the two surfaces and computing the Hausdorff distance between the two discrete sets of samples. The popular Metro tool [CRS98] super-samples one surface and computes the maximum of the distance between the samples and the other surface.

Because of the complexity of the computation of an exact discrepancy measure, most simplification algorithms use a local error estimation. Consider a vertex that has moved from its initial position  $v$  to a new position  $v'$ , as a result of a vertex clustering step or of a series of edge-collapses. The distance  $\|vv'\|$ , which is bounded by the cell diagonal in the vertex clustering approach, provides a conservative bound on the Hausdorff error resulting from this displacement. However, it is too conservative when the mesh is nearly planar in the vicinity of  $v$  and when the vector  $vv'$  is tangent to the surface. Clearly, we want to measure the component of the displacement of that vertex along the normal to the surface.

The error resulting from the collapse of a vertex  $v_1$  to its neighbor  $v_2$ , can be estimated by the dot-product  $v_1v_2 \cdot N_1$ , where  $N_1$  is the surface normal computed at  $v_1$ . Although simple, this formulation does not guarantee a conservative error bound. Ronfard and Rossignac [RoRo96] have used the maximum of the distance between the new position of  $v'$  and the planes that contain the original triangles incident upon  $v$ . The distance between  $v'$  and the plane containing vertices  $(v,a,b)$  is  $vv' \cdot (va \times vb) / \|va \times vb\|$ . The term  $(va \times vb) / \|va \times vb\|$  may be pre-computed and cached for each triangle in the original mesh using its vertices  $v$ ,  $a$ , and  $b$ . Note that for very sharp edge and vertex, an additional

plane is necessary to measure excursions of  $v'$  that would be tangential to all the faces incident upon  $v$  and yet would move away from the surface. That normal to that additional plane may be estimated by the surface normal estimation at  $v$ . The cost of this approach lies in the fact that, as vertices are merged through series of edge collapses, one needs to keep track of the list of all the planes that contain the triangles incident to these vertices in the original model. Furthermore, for each new edge-collapse candidate, the distance between the new position  $v'$  must be computed to all these planes. If the edge collapse is executed, the lists of planes must be merged.

Trading the conservative maximum error bound of [RoRo96] for a mean square measure, Garland and Heckbert [Garl98, GaHe98, Garl99] have drastically reduced the cost of maintaining error estimates. The square distance between point  $P$  and plane  $\text{Plane}(Q_1, N_1)$  through point  $Q_1$  with normal  $N_1$  is  $(N_1 \cdot Q_1 P)^2$ . It is a quadratic polynomial in the coordinates  $(x, y, z)$  of  $P$ . Hence, it may be written as:

$$D_1(P) = a_1 x^2 + b_1 y^2 + c_1 z^2 + d_1 yz + e_1 xz + f_1 xy + g_1 x + h_1 y + i_1 z + j_1.$$

Note that the sum of the squared distances from  $P$  to two planes,  $\text{Plane}(Q_1, N_1)$  and  $\text{Plane}(Q_2, N_2)$  is  $D_1(P) + D_2(P) = (a_1 + a_2)x^2 + (b_1 + b_2)y^2 + (c_1 + c_2)z^2 + (d_1 + d_2)yz + (e_1 + e_2)xz + (f_1 + f_2)xy + (g_1 + g_2)x + (h_1 + h_2)y + (i_1 + i_2)z + (j_1 + j_2)$ . Based on this observation, in a preprocessing stage, we can pre-compute these 10 coefficients  $(a_k, b_k, \dots, j_k)$  for each corner of each triangle. Then for each vertex  $v_m$ , we compute the 10 coefficients  $(a_m, b_m, \dots, j_m)$  by adding the respective coefficients of its corners. They define the function  $D_m$  associated with that vertex. During simplification, we can estimate the cost of an edge collapse that would move a vertex  $v_1$  to a location  $v$ , by  $D_1(v)$ . We always perform the collapse with the lowest estimate. When two vertices,  $v_1$  and  $v_2$ , are merged, the coefficients of the quadric error function of the combined vertex are the sums the corresponding coefficient of  $D_1$  and  $D_2$ .

Note that given such a quadratic function  $D$ , the location of the vertex that minimizes it may be estimated by solving the linear system of equations,  $2ax + ez + fy + g = 0$ ,  $2by + dz + fx + h = 0$ , and  $2cz + dy + ex + i = 0$ , that cancel the derivatives of  $D(P) = ax^2 + by^2 + cz^2 + dyz + exz + fxy + gx + hy + iz + j$  with respect to the three coordinates [RoRo96, GaHe98].

The FanGrower approach proposed here uses such a linear formulation to optimize the location of the tip of each cap. However, instead of the coefficient of the planes supporting the triangles of a cap, it uses planes defined by the vertices of the cap and the associated surface normals, which are estimated from neighboring triangles.

Kalvin and Taylor [KaTa96] estimate the flatness of their superfaces by considering the dual space of the plane parameters. To each vertex  $v$ , they associate a family of planes that are closer to  $v$  than some threshold. This family is represented by a convex set in dual space. A superface cluster is sufficiently flat if the intersection of these convex sets is not empty. Because the incremental evaluation of this intersection is computationally too expensive, Calvin and Taylor maintain an ellipsoid that fits inside it. Each time a triangle is added to a superface, they replace the old

ellipsoid  $E$  by a new one that fits inside the intersection of  $E$  with the slab of the new triangle. We have considered using a similar approach for testing whether a cap is a close approximation to a fan, but have concluded that using the slabs around the triangles of a cap, or even around the planes defined by the cap vertices and their normals is too conservative. Therefore, we have developed a new, more precise, conservative error measure, which for each vertex  $v$  of a cap, computes its distance to a particular pair of line segments at which the fan intersects a particular plane passing through  $v$  and the tip of the fan.

## Prior art in multi-resolution modeling

Many interactive 3D viewing systems subdivide the model into isolated objects (for example, the components of an assembly or the buildings of a city) and pre-compute a series of static approximations for each object [Sc&95]. Each approximation is obtained by applying a simplification process with a higher view-independent tolerance, and thus having a significantly lower triangle count than the previous one. At runtime, the rendering subsystem decides which approximation to use for each object, based on an estimate of the error associated with using a particular approximation [Funk93]. This process works well for complex scenes composed of relatively small objects. However, it is not effective for relatively large and complex objects, because, if a subset of the large object is close to the viewpoint, the entire object will have to be displayed at the highest resolution, even though the more distant portions could have been displayed much faster using a much lower resolution.

Several approaches [XiVa96, PuSc97, Hoppe97, LuEr97] pre-compute a single multi-resolution representation and use it to accelerate the derivation of a new simplification from the previous one. The simplification process creates a cluster-merging tree. Three items are associated with each node  $N$  of the tree: a simplification operation that merges the vertex-clusters of the children of  $N$ , the information needed to perform the inverse cluster-splitting operation, and an estimator of the error resulting from using the merged cluster instead of the split clusters. (Note that when simplification is restricted to preserve topology, the cluster hierarchy is a forest of trees. Allowing topological changes, as in [RoBo93, PoHo97, and GaHe97] makes it possible to group the forest into a single tree.) Xia and Varshney [XiVa96] and Hoppe [Hopp97] use a single edge-collapse as the individual cluster-merging step. Luebke and Erikson [LuEr97] claim more flexibility and for example propose to merge Rossignac and Borrel's cell-based clusters [RoBo93] into an octree. Lindstrom et al. [Lind96] use a similar quadtree-based scheme for an adaptive terrain model. In all these cases, the decision tree imposes a partial dependency order on the cluster merging operations: It is not possible to merge two arbitrary clusters unless one merges their common ancestor. A similar scheme is also discussed in [Ci&95, DePu92]. Thus, as the viewing conditions evolve, the cluster-merging or cluster-splitting operations are carried out one by one.

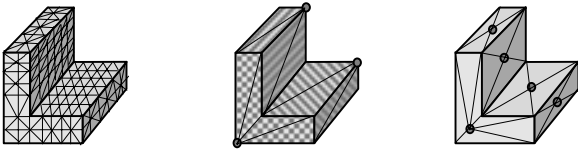
In contrast, FanGrower uses a representation that makes the dependency maintenance trivial and eliminates the run-time



cost of performing the cluster-merging and cluster-splitting operations. The mesh is represented by the crudest triangle mesh in which the frame tip-vertices and rim-edges are identified. One bit per tip, set by the viewing application, is used to identify the tip of frames that should be rendered as caps. All rim-edges of these frames are automatically set to be refined. When a rim-edge is refined, it is treated as a polyline. When it is simplified, it is replaced by an edge between the first and the last vertex of this polyline. Non-refined frames are rendered as fans implicitly defined by their tip and rim (which may include simplified beams or refined portions of the original rim). Thus, the tip-bits may be set independently of one another without any dependency constraints nor any cost associated with cluster merging or cluster splitting operations.

## Our contributions in the context of prior art

To place our approach in the broader context of prior art, consider that several successful simplification techniques (vertex decimation [Schr97], edge or vertex collapsing [PoHo97, GaHe97, RoRo96, Hopp96], and face merging [KaTa96]) all perform the following four tasks, although in different order: (1) identify vertex clusters, (2) coalesce each cluster into a single point (attractor), (3) compute an optimized position of these attractors, (4) remove triangles with more than one vertex in the same cluster, unless this would alter connectivity. For example, connected components of edge collapses define vertex clusters, independently of the order in which edge collapses were performed. Similarly, superfaces treat their internal vertices as a single cluster. FanGrower falls in this category.



**Fig 4:** The over-tessellated L-bracket (left) may be represented exactly using 3 fans (center) or 8 super-faces (right). The tips of the fans are shown as small circles.

In particular, our frames may be viewed as an extension of Kalvin and Taylor super-faces [KaTa96], which produced similar fan connectivity, but are restricted, by design, to be nearly flat. The flexibility of using non-flat fans reduces the number of clusters necessary to represent a shape, because caps may contain sharp corners or saddle-points at their tip. To appreciate the potential of fans, consider the contrived example of Fig. 4, where the L-shaped object shown below, which can be represented exactly with 3 fans, but would require 8 super-faces.

The novelty of our approach is based the following research contributions, detailed in the following sections:

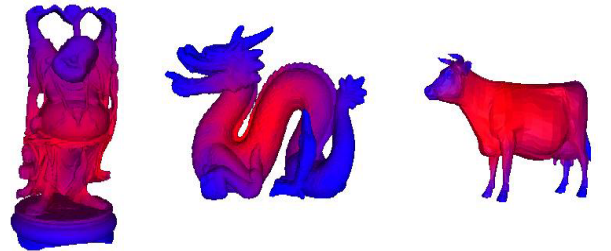
- The segmentation of the mesh into fan-like caps, rather than planar regions.

- The idea of using an isolation measure for the selection of the seed-tips vertices
- A geometric formulation of tests that guarantee a well shaped caps without undesirable rim undulations.
- An efficient computation of a tight bound on the Hausdorff error between the cap and its fan and frame.
- An efficient computation of an optimized location of the tip of a cap using a quadric error minimization.
- A compact data structure for storing fan-meshes and their refinements.
- The use of implicitly defined fans to ensure a water tight interface between caps and frames.

## Details of our approach

### Computing the seed-tips

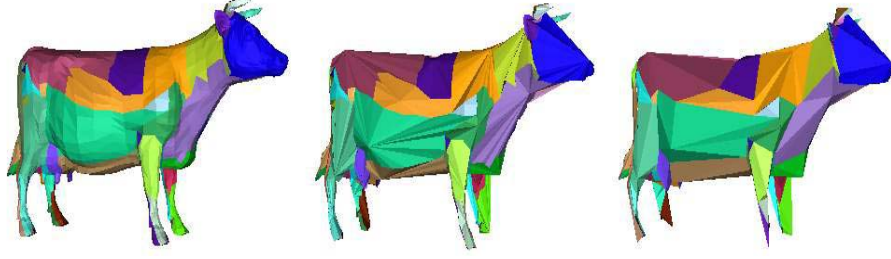
The growth of each cap starts by selecting the initial location of the tip as the point of maximal isolation. The **isolation** at a point P in a surface is measured by the average distance between P and all other points of the surface. Points on tips of long branches tend to correspond to local maxima of the isolation measure. We use a graph distance (with uniform edge weights) of the dual graph of the mesh to quickly estimate the isolation of each triangle. The details and speed-ups of this computation are discussed in [Hi&01]. As illustrated in Fig.5, the tips of elongated features correspond to the areas of higher isolation. Thus, we select as seed-tip, center of the most isolated triangle.



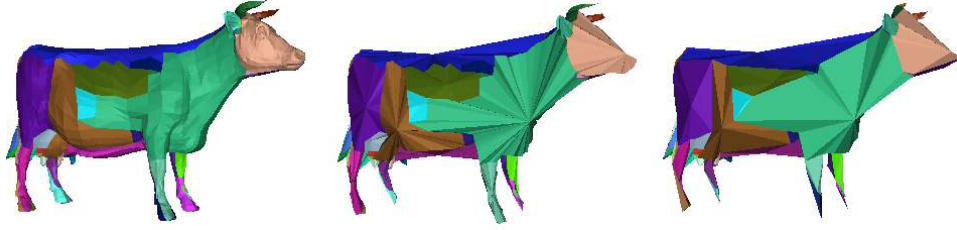
**Fig 5:** The blue regions are more isolated than the red ones. We seed the next cap at the most isolated place.

The cap is grown as explained below. When a cap can no longer be extended without exceeding a prescribed error, we start another cap at a new seed-tip, which is selected as the most isolated point of the remaining (virgin) portion of the surface.

We have explored two approaches for computing seed-tips for subsequent caps. The faster one re-uses the isolation measure computed initially. The slower, but better, one considers the boundary of the caps as zero ground and propagates the distance measure away from the caps along the surface. When more than one triangle has the highest distance, the dual graph of these is considered and the set of nodes that are the furthest away from the leaves of the graph is located. The new seed is the barycenter of one of these triangles. The results are compared in Fig 6a and 6b.



**Fig 6a:** The 5804 triangle mesh of the cow model was divided by into 126 caps (left). The fan mesh (center) contains 1936 triangles. The frames (right) contain a total of 678 triangles. The caps, fans, and frames were generated by FanGrower in 38 seconds. Here the approach to selects the next seed was based on computing isolation weight once initially.



**Fig 6b:** The same cow model was divided by into 116 caps (left). The fan mesh (center) contains 1702 triangles. The frames (right) contain a total of 592 triangles. The caps, fans, and frames were generated by FanGrower in 40 seconds. Here the approach to selects the next seed was based on computing the triangle that is furthest from the boundary of already visited triangles.

### Growing the cap

The cap is initialized to be the virgin triangle containing the seed-tip. (Note that at first sight, it may appear more strategic to pick the seed tip at a vertex of a virgin triangle, because the initial cap made of all of the incident virgin triangles can be perfectly approximated by the corresponding fan. However, in order to simplify implementation, we did not want to distinguish between cases where the tip would be surrounded by virgin triangles and where it would lie on the rim of a cap. So, we have decided to select seed-tips at the center of the most isolated virgin triangle. In practice, this choice has little consequences, because the tip would be moved automatically to a nearby vertex if necessary.)

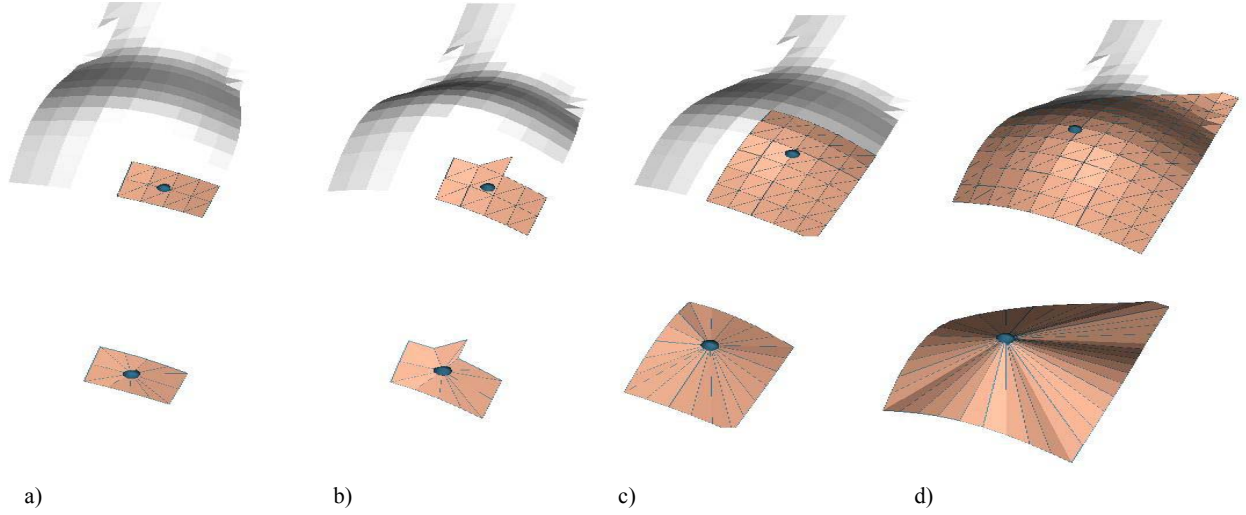
The candidate virgin triangles considered for inclusion in a cap are considered in an order that spirals around the seed-triangle. We have used a slightly modified EdgeBreaker traversal [SaRo02] for accessing them.

Because we want the caps to remain simply connected, we reject candidate triangles that would produce a self-intersection in the rim of the current cap. Note that they are easily detected by the Edgebreaker traversal (as S-cases in

the Edgebreaker terminology), without the need for maintaining any explicit representation of the rim.

We also reject triangles that would disturb the monotonicity of the rim's circular progression around an axis of rotation, defined later, which we associate with the cap and use for computing a tight error bound.

While growing the cap, the tip may be recomputed and moved to a new optimized position. To minimize the amount of work, we first accept all triangles that can be accepted without moving the tip. Then, we temporarily include other candidate triangles, one at a time, and for each one, if necessary, recompute the tip and evaluate the resulting error between the cap and the fan. We also check that the rim of the tentatively extended cap satisfies all the requirements with respect to the new tip. If the error does not exceed the threshold and the rim requirements are satisfied, we keep the candidate triangle in the cap and keep the new location of the tip. Otherwise, we do not absorb the new triangle and go back to the previous tip location. The process is illustrated in Fig. 7.

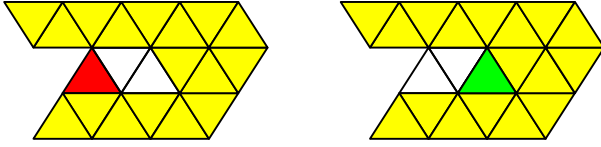


**Fig 7.** The cap (top) and its fan (bottom) during the cap-growing process. a) An initial cluster could be grown without updating the seed-tip. b) The inclusion of a new triangle requires updating the tip. c) The region is grown further by a sequence of triangle inclusions and occasional adjustments of the tip. d) Now more triangles can be added to the final cluster.

Before accepting a candidate triangle into the cluster, we perform the following tests.

#### Ensuring that the cap remains simply connected

**S-test:** Because we want the cap to be simply connected, we do not accept triangles that have 3 of their vertices and only one of their edges on the current rim (see Fig. 8). These correspond to S triangles in the Edgebreaker terminology.



**Fig. 8:** The red candidate triangle (left) is rejected, because its inclusion would make the yellow cap not simply-connected. Note that it has one edge and 3 vertices on the rim. The green triangle (right) passes the S-test.

One of our technical contributions is an efficient estimation of the error bound, which resulted from an extensive evaluation of several alternatives. The solution we have adopted is based on maintaining two things during the cap-growing process: a normal direction at the tip and monotonicity property of the rim. We discuss them first.

#### Computing the cap normal

A normal  $N$  at each tip is estimated as the weighted average of the normals of its fan triangles. The weights are proportional to the areas of the triangles. Thus,  $N$  is simply computed as the vector sum of all the cross products of pairs of vectors from the tip to consecutive vertices along the rim. Then  $N$  is divided by its length to produce a unit normal vector.

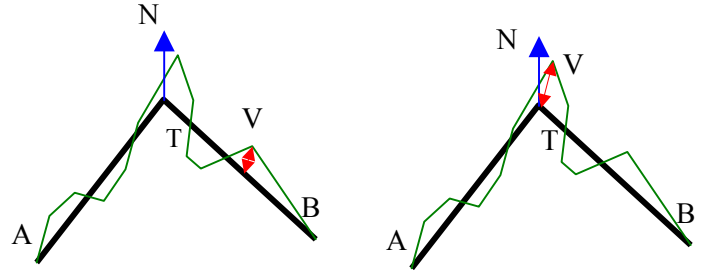
#### Preventing rim undulations

Now, consider the plane  $P$  through the tip  $T$  orthogonal to  $N$ . We say that the rim is **monotonic**, if its projection onto  $P$  is star-shaped as seen from  $T$ . Thus, a rim is monotonic if

for each triplet  $(A,B,C)$  of its consecutive vertices, the scalars  $N \cdot (TA \times TB)$  and  $N \cdot (TB \times TC)$  have same sign. By construction, we ensure that all caps have a monotonic rim.

#### Computing the error bound

Let  $V$  be a vertex of the cap and  $Q$  a plane parallel to  $N$  and passing through  $V$  and  $T$ .  $Q$  intersects the rim at two points,  $A$  and  $B$  and intersects the fan in two line segments,  $TA$  and  $TB$  (See Fig. 9). We report the minimum distance between  $V$  and the union of  $TA$  with  $TB$ . The maximum of these reported distances, for all vertices of the cap, is a bound on the Hausdorff distance between the cap and its fan.



**Fig 9:** Consider the plane  $Q$  passing through tip  $T$  and a vertex  $V$  of the cluster and parallel to the normal  $N$ .  $Q$  intersects the cap at two line segments (thick curves), which join  $T$  to the intersections  $A$  and  $B$  of  $Q$  with the rim. The minimum distance between  $V$  and the two line-segments may occur inside a line segment (left) or at  $A$ ,  $B$ , or  $T$  (right)

#### Optimizing the location of the tip

When the error resulting from incorporating a new triangle exceeds the given threshold and not a single virgin triangle may be attached without moving the tip, we tentatively adjust the tip and its normal to the tentatively extended cluster and accept the triangle if the resulting error is within tolerance. Several of the approaches that we have considered for computing the position of the tip for which the Hausdorff error, or even its bound, would be minimized are computationally too expensive for practical purposes.

The solution, which we have retained for FanGrower adjusts the **location of the tip** using a linear expression discussed below that minimizes Garland and Heckbert’s quadric error,  $D$ . In its original formulation, the quadric error represents the sum of the squared distances between a point and a set of planes. The same expression has been used previously for computing the optimal position of a cluster of vertices collapsed into a single point through a sequence of edge-collapses [RoRo96, GaHe97]. Thus, we could envision using the planes that contain the faces of a cap to define the tip through this quadric error minimization. However, to reduce the bias possibly produced by large collections of small faces not properly aligned with the fan, we do not use the planes through the triangles of the cap, but instead use the planes that each pass through a different vertex of the cap and are orthogonal to the corresponding vertex normal. This approach is illustrated in Fig. 10.



**Fig 10:** A cross-section through a cap shaped like a wedding cake is shown as a thick staircase pyramid. Each edge represents a triangle of the cap. Using the planes that support the triangles (thin vertical and horizontal lines, left) and minimizing the square of the distances to all supporting planes yields a tip (large dot left) below the center of the cake, because the bottom horizontal annuli of the cake have a larger surface than the top ones, and hence yield superior weights for the bottom plane. Instead of the planes containing the triangles, FanGrower uses planes through the vertices of the cap. Each plane is perpendicular to the surface normal estimated at the vertex it stabs. These planes follow the conical structure of the pyramid and yield a much better location for the tip (right).

### Simplifying the rim

Once the caps cover the entire mesh, we can replace each cap by its fan. However, the rims of the fans may contain unnecessarily large numbers of vertices. We simplify them as follows. A vertex of the mesh that is on the rim of three or more clusters is called a **metavertex**. Removing the metavertices from the rims decomposes them into imply connected polylines. We simplify each one of them independently using the 3D version of the Douglas-Peucker line simplification algorithm [DoPe73]. The simplification of each run between metavertices A and B starts by finding the vertex M in the run that is the furthest from the line-segment AB. If the distance from M to AB exceeds the desired threshold,  $E$ , we split the run at M and iterate the process on the two parts of the run: AM and MB. The simplification of each run yields a series of one or more line segments, which we call **beams**. Thus, the rim of each cap is approximated by a connected cycle of beams. The frame,

which is a simplified representation of the cap, is defined by the series of triangles that each connects the tip of the cap to one of its beams.

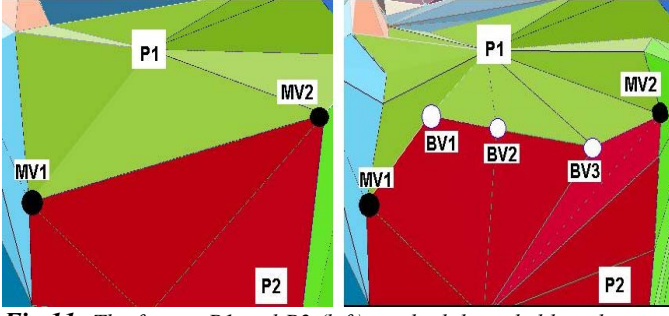
Note that we ensure that the metavertices are preserved through this rim-simplification process.

Note also that the vertex-to-edge distance used in our rim-simplification process is overly conservative. In fact, it could be relaxed in the areas of the rim where the two incident fans form a shallow angle. We have chosen not to do so, because we want to be able to **guarantee a bound on the Hausdorff error between the cap and its frame**. Indeed, during the cap-growing process, we have ensured that the distance between a cap and its initial (not simplified) fan is less than some prescribed error bound  $E$ . The rim simplification described in this section ensures a Hausdorff error of less than  $E$  between a rim and its simplification. As a consequence, the Hausdorff error between the fan and the corresponding frame is also less than  $E$ . As a proof, consider a continuous “flattening” process that deforms a fan into its frame, by moving the rim vertices to their closest counterpart on the simplified rim. During this process, no vertex has moved by more than  $E$ . Given that the tip has not moved, no point of any of the triangles has moved by more than  $E$ . This proves that all points on the fan are within  $E$  from the frame. To prove the reverse statement, associate with each edge of a run its orthogonal projection on the beam (defined by the segment of the beam joining the two points of the beam that are the closest to the edge). The union of these projections covers the beam. Furthermore, each projection lies within distance  $E$  of its edge. This may be proven by considering that the set of all points that lie within distance  $E$  from an edge is a convex set and that, because it contains the end-points of the projection of the edge, it must also contain the entire projection. Consequently, the Hausdorff distance between a cap and its frame is less than  $2E$ .

### Representation of the bi-resolution model

We have designed a compact representation of this bi-resolution model that facilitates the independent refinement of each frame to its full resolution cap. The model is represented by the crude triangle mesh in which a subset of the vertices are identified as tips of frames. This is the crudest level of detail. With each tip and with each beam-edge that is not incident upon a tip, we associate a refinement. Edge-refinements specify an ordered sequence of interior vertices, which have been removed during rim simplification. The refinement associated with a tip specifies the connectivity of the cap and the location of its interior vertices. At any given moment, one bit per tip defines which frame is refined. The edges that bound a refined frame are refined automatically. Thus, frames that are not selected for refinement, but bound refined frames, have some or all of their bounding edges refined. These frames would be rendered as partially refined fans as shown Fig. 11.

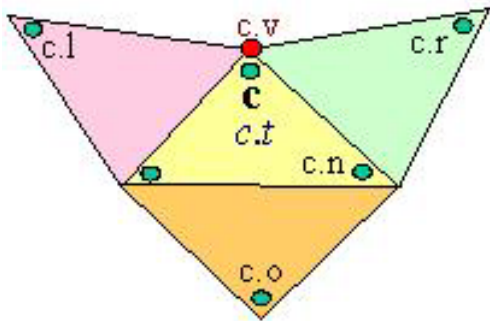




**Fig 11:** The frames  $P1$  and  $P2$  (left) are both bounded by a beam joining the metaverices  $MV1$  and  $MV2$ . When one of the frames needs to be rendered as a cap, the beam is marked as refined and automatically replaced (right) by the polyline ( $MV1$ ,  $BV1$ ,  $BV2$ ,  $BV3$ ,  $MV2$ ).

To support this independent resolution selection, we re-order the vertex table to first list the tip vertices, then the metaverices, then the other rim vertices (grouped per beam and ordered along the rim), and finally the remaining vertices (grouped per cap). We also re-order the incidence table, listing first the triangle/vertex incidence for the triangles joining the tip of each frame to its beams. Then, we list the incidence tables of each cap.

We use a Corner Table [RSS01, RSS02] to represent the triangle/vertex incidence and triangle/triangle adjacency information as two arrays of integers. In the Corner Table, triangle-vertex **incidence** defines each triangle by the three integer references to its vertices. These references are stored as **consecutive** integer entries in the **V table**. Note that each one of the  $3T$  entries in **V** represents a **corner** (association of a triangle with one of its vertices). Let  $c$  be such a corner. Let  $c.t$  denote its triangle and  $c.v$  its vertex. Remember that  $c.v$  and  $c.t$  are integers in  $[0, V-1]$  and  $[0, T-1]$  respectively. Let  $c.p$  and  $c.n$  refer to the **previous** and **next** corner in the cyclic order of vertices around  $c.t$ . Although the **V table** suffice to completely specify the triangles, it does not offer direct access to a neighboring triangle or vertex. We use the reference to the **opposite** corner,  $c.o$ , which we cache in the **O table** to accelerate mesh traversal from one triangle to its neighbors. For convenience, we also introduce the operators  $c.l$  and  $c.r$ , which return the **left** and **right neighbors** of  $c$  (Fig. 12).



**Fig 12:** Corner operators for traversing a corner table representation of a triangle mesh.

Note that we do not need to cache  $c.t$ ,  $c.n$ ,  $c.p$ ,  $c.l$ , or  $c.r$ , because they may be quickly evaluated as follows:  $c.t$  is the integer division  $c.t \text{ DIV } 3$ ;  $c.n$  is  $c-2$ , when  $c \text{ MOD } 3$  is 2, and  $c+1$  otherwise; and  $c.p$  is  $c.n.n$ ;  $c.l$  is  $c.n.o$ ; and  $c.r$  is  $c.p.o$ . Thus, the storage of the connectivity is reduced to the **O** and **V** arrays.

We assume that all triangles have been consistently **oriented**, so that  $c.n.v = c.o.p.v$  for all corners  $c$ . For example, one may adhere to the convention that when a triangle  $c.t$  is visible by a viewer outside of the solid (i.e., the finite set that is bounded by the triangle mesh), the three vertices,  $c.p.v$ ,  $c.v$ , and  $c.n.v$ , appear in clockwise order.

The top of the corner table references only tip and metaverices and represents the lowest level resolution of the model, i.e. its frames. We order the corners of these frame triangle so that their first corner listed references the tip vertex. We pick one triangle in each frame and list them first, in the same order as we list the tip vertices.

Because of this re-ordering, we are able to use a simple array that associates with each frame: a bit, indicating whether it should be rendered as a cap or not, and the starting and ending index to the contiguous sequence of entries of the corner table that describe the connectivity of the triangles in the corresponding cap.

We also use an array **B** that associates three entries with each beam: (1) a bit indicating whether it needs to be refined by inserting the vertices of the portion of the rim that the beam approximates, (2) an integer reference to the first vertex of the sequence, and (3) an integer reference to the last vertex of the sequence. The entries of **B** are listed in the same order as the frame-triangles, and thus may be accessed without having to store an explicit reference to them.

### A comparison of Fan-Growing strategies

Inspired by the Hierarchical Face Clustering approach developed by Garland et.al. [Ga&01], we have initially considered a hierarchical, bottom-up construction of the caps. This approach starts by selecting a set of vertices so that each triangle is incident upon at least one vertex of the set and the number of triangles incident upon more than one is minimized. Then, for each edge between two caps, say caps  $C_1$  and  $C_2$ , we consider merging the two caps. As pointed out in [Ga&01], this operation is analogous to an edge-collapse of the dual graph of the cap connectivity graph. Following the edge-collapse strategy of [Hopp96], the error resulting from the collapse of each edge in the dual graph can be maintained in a priority queue and the edge with the lowest error collapsed first. We have concluded that this bottom-up process makes decisions early on that cannot be undone later, and hence lacks the granularity necessary for growing the caps that satisfy both the error bond and the shape constraints necessary for an efficient computation of the maximum error bound.

We have also considered using the flatness estimate proposed in [Ga&01], which is defined as the average squared distance of all the points in a cluster to a plane. The plane is the least squares best-fit plane to these points. This approach differs from the quadric-based error used for

simplification [GaHe97] in that the quadric error is computed by summing over a set of points with a fixed normal rather than a set of normals with a fixed point. An additional cost measure, which takes into account the ratio between the squared perimeter and the area of the cluster, is used in [Ga&01] to favor nicely shaped clusters. This approach focuses on identifying nearly flat clusters and, according to the authors, is not used to alter the original surface geometry in any way, nor does it produce any new approximate surfaces. In contrast, FanGrower uses clustering to identify portions of the original mesh that may be approximated by fans within a guaranteed maximal error tolerance.

## Results

The statistics that we have obtained by running FanGrower on several popular models are shown in Fig. 1, 6a, 6b, 13 and 14.

Consider that a fan may be produced by collapsing all the internal (non-rim) edges of a cap. Furthermore, the frames may be obtained by collapsing some of the edges of the rim. Consequently, the FanGrower algorithm may be viewed as a sequence of edge-collapses constrained to produce the desired fan-like structure. Because of the various constraints imposed during the formation of the caps, constraint, we should expect the results produced by FanGrower will require a higher error tolerance than would an unconstrained edge-collapse simplification to achieve the same reduction in triangle count. To quantify this overhead, we have compared the accuracy of the frame triangulations produced by FanGrower to the simplified models produced by Qslim [Qslim] through sequences of edge-collapses. We ensured that both simplified models had nearly similar numbers of triangles. We used Metro [Metro] to measure the maximum discrepancy. Our experiments show that the

error of the simplified models produced by Qslim is 50 to 80% lower than the one produced by FanGrower.

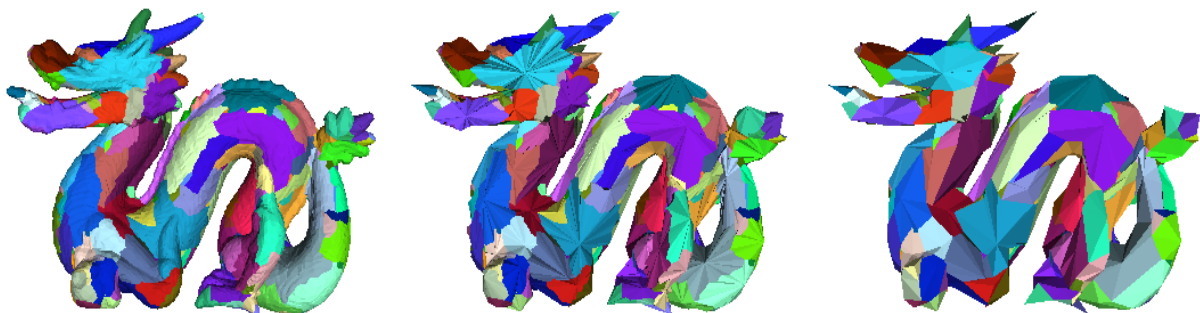
We argue that, one may be willing to trade this loss of accuracy for the flexibility and simplicity of the bi-resolution representation offered by our fan-meshes. The fan-mesh structure proposed here requires a very small overhead over the representation of the original mesh, and may thus be preferred to a full-fledged multi-resolution representation in situations where a high granularity of levels-of-detail is not required.

## Conclusion

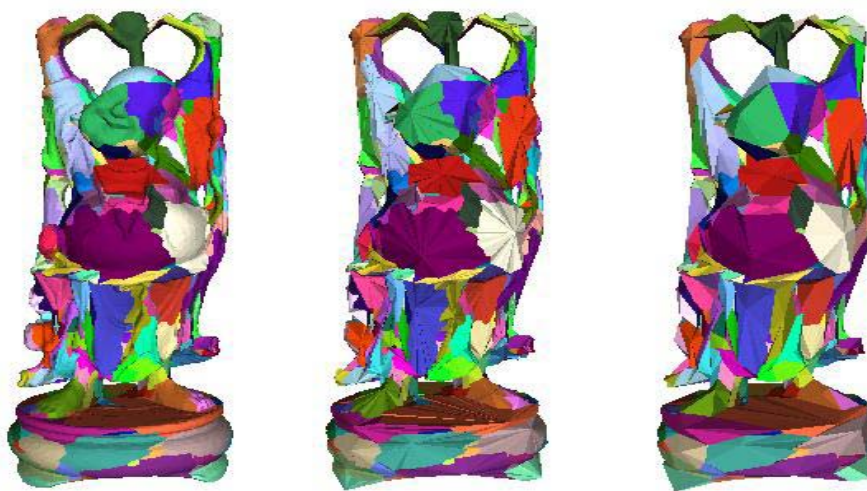
We have proposed an automatic technique that segments a manifold triangle mesh into regions, called caps, associates with each cap a simplified triangle fan, called frame. By reordering the triangle-mesh representation of the original model and by complementing it with a description of the triangulation of the frames and by additional references and bit-masks, we obtain a compact, yet flexible, bi-resolution representation, which may be arbitrarily refined by designating which fans should be rendered as full-resolution caps. We guarantee a water tight mesh connectivity.

## Acknowledgements

This project was partly supported by a DARPA/NSF CARGO grant #0138420. The authors thank Greg Turk, Eugene Zhang, and Andrzej Szymczak for their comments and suggestions. In particular, various clustering strategies from which the proposed approach has emerged have been explored in an ongoing collaboration with Greg Turk and Eugene Zhang.



**Fig 13:** The 20000 triangle mesh of the dragon (courtesy of [NoTu99]) was divided by into 410 caps (left). The fan mesh (center) contains 5888 triangles. The frames (right) contain a total of 2212 triangles. The caps, fans, and frames were generated by FanGrower in 188 seconds.



**Fig 14:** The 20000 triangle mesh of the happy Buddha (courtesy of [NoTu99]) was divided by into 502 caps (left). The fan mesh (center) contains 6146 triangles. The frames (right) contain a total of 2744 triangles. The caps, fans, and frames were generated by FanGrower in 197 seconds

## REFERENCES

- [An&96] C. Andujar, D. Ayala, P. Brunet, R. Joan-Arinyo, J. Sole, Automatic generation of multi-resolution boundary representations, Computer-Graphics Forum (Proceedings of Eurographics'96), 15(3):87-96, 1996.
- [Blak87] E. Blake, A Metric for Computing Adaptive Detail in Animated Scenes using Object-Oriented Programming, Proc. Eurographics'87, 295-307, Amsterdam, August 1987.
- [Ci&95] P. Cignoni, E. Puppo and R. Scopigno, Representation and Visualization of Terrain Surfaces at Variable Resolution, Scientific Visualization 95, World Scientific, 50-68, 1995.  
<http://miles.cnuce.cnr.it/cg/multiresTerrain.html#paper25>.
- [Co&96] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F. and Wright, W., Simplification envelopes. In Computer Graphics Proc., Annual Conf. Series (Siggraph '96). ACM Press.
- [CRS98] P. Cignoni, C. Rocchini and R. Scopigno, "Metro: measuring error on simplified surfaces", Proc. Eurographics '98, vol. 17(2), pp 167-174, June 1998.
- [DePu92] L. De Floriani, E. Puppo, A hierarchical triangle-based model for terrain description, in Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, Ed. A. Frank, Springer-Verlag, Berlin, pp. 36--251, 1992.
- [DoPe73] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10:112-122, 1973.].
- [Funk93] T. Funkhouser, C. Sequin, Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments, Computer Graphics (Proc. SIGGRAPH '93), 247-254, August 1993.
- [Ga&01] M. Garland, A. Willmott, and P. Heckbert. Hierarchical Face Clustering on Polygonal Surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics*, March 2001
- [GaHe97] M. Garland and P. Heckbert, Surface simplification using quadric error metrics, Proc. ACM SIGGRAPH'97. pp. 209-216. 1997.
- [GaHe98] M. Garland and P. Heckbert. Simplifying Surfaces with Color and Texture using quadric Error Metric. Proceedings of IEEE Visualization, pp. 287-295, 1998.
- [Garl98] Michael Garland. Quadric-based Polygonal Surface Simplification. PhD thesis, Carnegie Mellon University, 1998.
- [Garl99] M. Garland, QSLim 2.0 [Computer Software]. University of Illinois at Urbana-Champaign, UIUC Computer Graphics Lab, 1999.  
<http://graphics.cs.uiuc.edu/~garland/software/qlim.html>.
- [He&96] T. He, A. Varshney, and S. Wang, Controlled topology simplification, IEEE Transactions on Visualization and Computer Graphics, 1996.
- [HeGa94] P. Heckbert and M. Garland, Multiresolution modeling for fast rendering, Proc Graphics Interface'94, pp:43-50, May 1994.
- [HeGa97] Paul Heckbert and Michael Garland. Survey of polygonal simplification algorithms. In Multi-resolution Surface Modeling Course. ACM SIGGRAPH Course Notes, 1997.
- [Hi&01] M. Hilaga, Y. Shinagawa, T. Kohmura, T.L. Kunii, "Topology Matching for Fully Automatic Similarity Estimation of 3D Shapes", Computer&Graphics, Proceeding of SIGGRAPH 2001, Los Angeles, 2001.]
- [Ho&93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh optimization," in Computer Graphics: Siggraph '93 Proceedings, 1993, pp. 19-25.
- [Hopp96] H. Hoppe, "Progressive meshes," Computer Graphics, vol. 30, no. Annual Conference Series, pp. 99-108, 1996.
- [Hopp97] H. Hoppe, View Dependent Refinement of Progressive Meshes, Proceedings ACM SIGGRAPH'97, August 1997.
- [Hopp98] H. Hoppe, "Efficient implementation of progressive meshes," Computers and Graphics, vol. 22, no. 1, pp. 27-36, 1998.
- [KaTa96] A.D. Kalvin and R.H. Taylor, "Superfaces: Polygonal mesh simplification with bounded error". IEEE



Computer Graphics and Applications, 16(3), pp. 64-67, 1996.

[Lind00] P. Lindstrom, Out-of-core simplification of Large Polygonal Models. Proc. ACM SIGGRAPH, pp. 259-262, 2000.

[Lind96] P. Lindstrom, D. Koller and W. Ribarsky and L. Hodges and N. Faust G. Turner, Real-Time, Continuous Level of Detail Rendering of Height Fields, SIGGRAPH '96, 109--118, Aug. 1996.

[LoTa97] K-L. Low and T-S. Tan, Model Simplification using Vertex-Clustering, Proc. 3D Symposium on Interactive 3D Graphics, pp. 75-81, Providence, April 1997.

[Lu&02] D. Luebke, M Reddy, J. Cohen, A. Varshney, B. Watson, R. Hubner, "Levels of Detail for 3D Graphics", Morgan Kaufmann, 2002.

[LuEr97] Luebke, D. and Erikson, C., View-dependent simplification of arbitrary polygonal environments. In ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97), 1997, pp. 199-208.

[Metro] P. Signoni, G. Impoco, Metro: measuring distances between surfaces *Computer Graphics Forum*, Blackwell Publishers, vol. 17(2), June 1998, pp 167-174

[MiSu95] J.S.B, Mitchell, and S. Suri, Separation and approximation of polyhedral objects, Computational Geometry: Theory and Applications, 5(2), pp. 95-114, September 1995.

[NoTu99] F.S. Nooruddin, Greg Turk Simplification and Repair of Polygonal Models Using Volumetric Techniques

[PoHo97] J. Popovic and H. Hoppe, Progressive Simplicial Complexes, Proceedings *ACM Siggraph'97*, pp. 217-224, August 1997.

[PuSc97] E. Puppo and R. Scopigno, Simplification, LOD and multiresolution: Principles and applications, Tutorial at the Eurographics'97 conference, Budapest, Hungary, September 1997.

[Qslim]

<http://graphics.cs.uiuc.edu/~garland/software/qslim.html>

[RoBo93] J. Rossignac and P. Borrel, "Multi-resolution 3D approximations for rendering complex scenes", Geometric Modeling in Computer Graphics, Springer Verlag, Berlin, pp. 445-465, 1993.

[RoCa99] J. Rossignac and D. Cardoze, "Matchmaker: Manifold Breps for non-manifold r-sets", Proceedings of the ACM Symposium on Solid Modeling, pp. 31-41, June 1999.

[RoRo96] R. Ronfard and J. Rossignac, "Full range approximation of triangulated polyhedra", Proc. Eurographics 96, 15(3), pp. 67-76, 1996.

[Ross95] J. Rossignac, Geometric Simplification, in Interactive Walkthrough of Large Geometric Databases (ACM Siggraph Course Notes 32), pp. D1-D11, Los Angeles, 1995.

[Ross96] J. Rossignac, Geometric Simplification, in *Interactive Walkthrough of Large Geometric Databases*, ACM Siggraph Course notes 35, pp. D1-D37, New Orleans, 1996.

[Ross00] J. Rossignac, "3D Compression and progressive transmission" Lecture at the ACM SIGGRAPH conference July 2-28, 2000.

[RSS01] J. Rossignac, A. Safonova, and A. Szymczak, "3D Compression Made Simple: Edgebreaker on a Corner-Table", Invited lecture at the Shape Modeling International Conference, Genoa, Italy, May 2001.

[RSS02] J. Rossignac, A. Safonova, A. Szymczak "Edgebreaker on a Corner Table: A simple technique for

representing and compressing triangulated surfaces", in Hierarchical and Geometrical Methods in Scientific Visualization, Farin, G., Hagen, H. and Hamann, B., eds. Springer-Verlag, Heidelberg, Germany, to appear in 2002.

[SaRo02] A. Safonova and J. Rossignac, Source code for an implementation of the Edgebreaker compression and decompression [www.gvu.gatech.edu/~jarek/edgebreaker/eb](http://www.gvu.gatech.edu/~jarek/edgebreaker/eb)

[Sc&92] W. Schroeder, J. Zarge, and W. Lorensen, Decimation of triangle meshes, Computer Graphics, 26(2):65-70, July 1992.

[Sc&95] B.-O. Schneider, P. Borrel, J. Menon, J. Mittleman, J. Rossignac, "BRUSH as a Walkthrough System for Architectural Models", Proc. 5th Eurographics Workshop on Rendering, Darmstadt (Germany), June 1994. In *Rendering Techniques'95*, Springer-Verlag, 389-399, New York, 1995.

[Schr97] W. Schroeder, A topology modifying Progressive Decimation Algorithm, in *Multiresolution Surface Modeling Course*, ACM Siggraph Course notes 25, Los Angeles, 1997.

[Turk92] G. Turk, "Retiling polygonal surfaces", Proc. ACM Siggraph 92, pp. 55-64, July 1992.

[Vars94] A. Varshney. "Hierarchical Geometric Approximations". PhD Thesis. *Department of Computer Science, University of North Carolina-Chapel Hill, USA*, 1994.

[XiVa96] J. Xia and A. Varshney, Dynamic view-dependent simplification for polygonal models, Proc. Vis'96, pp. 327-334, 1996.